INFORMATION AND CONTROL **29**, 243–251 (1975)

On the Termination of Program Schemas

A. J. KFOURY

Massachusetts Institute of Technology, Cambridge, Massachusetts

AND

D. M. R. PARK

University of Warwick, Coventry, England

We show in this article that program schemas terminate because of second-order reasons, in a sense to be made precise.

It is an undecidable question whether a program schema terminates for a given interpretation.

We show here that, in all (nontrivial) cases, if a program schema terminates it cannot be because of first-order conditions, written in terms of the relation and function symbols appearing in the program schema itself. On the other hand, it is not difficult to establish that these conditions can be expressed as countably infinite disjunctions of first-order quantifier-free sentences and thus, *a fortiori*, in a second-order language.

As a consequence, the undecidability of the forementioned question cannot be reduced to that of a set of first-order sentences, in a sense to be further specified.

Part of the novelty in the present note is the use of model-theoretic techniques to obtain results about program schemas.

1. DEFINITIONS

Program schemas are abstract representations of computer programs. Given a countable set of variables (denoted by x_0 , x_1 , x_2 ,...), a set of basic operations

(denoted by $f_0, \ldots, f_m$) which may be used to assign new values to the variables, and a set of basic relations (denoted by $r_0, \ldots, r_l$), a program schema specifies the order in which computations involving the basic operations are to be performed in terms of truth values taken on by the basic relations. We call the finite sequence

$$\langle r_0, \ldots, r_l, f_0, \ldots, f_m \rangle$$

the *similarity type* of the program schema.

Several classes of program schemas have been defined and studied in the literature, the most familiar one being the class of flow-chart schemas. Although our results are true of any class of program schemas which satisfies a mild condition (this point is taken up again in Section 5), we shall restrict our attention, for definiteness, to the class of flow-chart schemas. Flow-chart schemas are defined in Luckham, Park, and Paterson (1970). They are program schemas which are representable by finite graphs, namely "flow-charts." The following is an example of one.

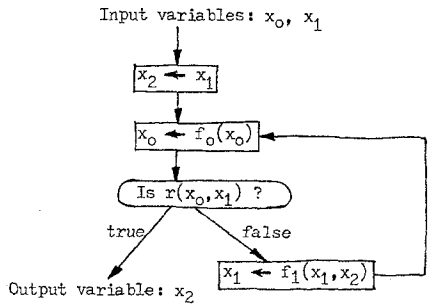

FIG. 1. A typical flow-chart schema.

Each of the rectangular boxes contains an *assignment* instruction, while the box with round edges contains a *test* instruction. The similarity type of the above flow-chart schema is $\langle r, f_0, f_1 \rangle$. If the symbols in the similarity type are given a particular interpretation, as a relation and operations on a given domain A, the schema becomes a program which can be executed by an idealized computer and which defines a function F from $A \times A$ to A. Starting with the first instruction, i.e., "$x_2 \leftarrow x_1$," and with an initial value from $A \times A$ assigned to (x_0, x_1), a computation proceeds sequentially by executing the instructions in the order in which they occur in the flow-chart. If, for example, the domain A is that of the natural numbers and r, f_0, and f_1 are,

respectively, the equality relation, the successor operation, and addition, then the function F which has been thus programmed is the following one:

$$F(a, b) = \begin{cases} b, & \text{if } (\exists n)(a + n = b \times n) \\ \text{diverging}, & \text{otherwise} \end{cases}$$

for all natural numbers $a, b \in A$. Note that, in the course of defining the semantics of the schema, we have also constructed a relational structure $\mathfrak{A} = \langle A, =, \times, + \rangle$.

Henceforth, whenever we use the terms "schema" or "program schema" we shall mean "flow-chart schema," unless otherwise specified. A remark about notation: Schemas will be denoted by capital S's; relational structures will denoted by German script letters, $\mathfrak{A}$, $\mathfrak{B}$, $\mathfrak{C}$,..., and their domains by the corresponding Latin letters, A, B, C,... .

A schema S interpreted in a structure $\mathfrak{A}$ will be called a *program*, which will be denoted by $S^{\mathfrak{A}}$. It is clear that the similarity type of S must be included in that of $\mathfrak{A}$, so that every symbol appearing in S may be properly interpreted. $S^{\mathfrak{A}}$ defines a partial function on the domain A, written in terms of the underlying relations and operations of $\mathfrak{A}$. An *interpretation* for S consists of a structure $\mathfrak{A}$ together with input values $\mathbf{a} = (a_1, \ldots, a_k)$ from the domain A of $\mathfrak{A}$, where k is the number of input variables of S. An interpretation, in symbols $\langle \mathfrak{A}, \mathbf{a} \rangle$, thus specifies a computation by S, denoted by $S^{\mathfrak{A}}(\mathbf{a})$, which is the sequence of instructions executed when values $a_1, \ldots, a_k$ from A are assigned to the input variables of program $S^{\mathfrak{A}}$.

A schema without loops will be called *trivial*. It is clear that a computation on a trivial schema always terminates, a diverging computation being possible only in the presence of loops.

2. The Main Question

Given a program schema S, can we formulate a set Φ of first-order conditions such that, given any interpretation $\langle \mathfrak{A}, \mathbf{a} \rangle$ for S, Φ is true of $\langle \mathfrak{A}, \mathbf{a} \rangle$ if and only if $S^{\mathfrak{A}}(\mathbf{a})$ terminates?

In this formulation, the similarity type of $\langle \mathfrak{A}, \mathbf{a} \rangle$ being countable, the first-order language in which Φ is written is also countable, i.e., it contains countably many distinct symbols, which is the language required to talk about $\langle \mathfrak{A}, \mathbf{a} \rangle$. We also make the requirement that Φ have a similarity type identical to, or contained in, that of S; that is, we do not allow Φ to involve operations and relations that are not used by S.

It is easy to give a program schema, namely a trivial one, so that the answer to the above question is in the affirmative. In fact, it turns out that if the answer is in the affirmative, then the given schema S is replaceable by a trivial one (for all the interpretations which are satisfying Φ). More specifically, we shall establish the following result [$*$]:

$$[*]\quad \begin{cases} \text{Given a schema } S, \text{ given any (possibly infinite) set } \Phi \text{ of first-order} \\ \text{sentences, if} \\ \qquad \text{for all } \langle \mathfrak{A}, \mathbf{a}\rangle, \langle \mathfrak{A}, \mathbf{a}\rangle \models \Phi \text{ implies that } S^{\mathfrak{A}}(\mathbf{a}) \text{ terminates,} \\ \text{then there exists a trivial schema } S_0 \text{ such that} \\ \qquad \text{for all } \langle \mathfrak{A}, \mathbf{a}\rangle, \langle \mathfrak{A}, \mathbf{a}\rangle \models \Phi \text{ implies that } S_0{}^{\mathfrak{A}}(\mathbf{a}) = S^{\mathfrak{A}}(\mathbf{a}). \end{cases}$$

In other words, we cannot constrain the interpretations for a schema S by first-order conditions in order to ensure termination—unless we also make superfluous the presence of the loops in S. An alternative statement for [$*$] is:

$$[**]\quad \begin{cases} \text{Given a schema } S, \text{ given any (possibly infinite) set } \Phi' \text{ of first-order} \\ \text{sentences, if} \\ \qquad \text{for all } \mathfrak{A},\ \mathfrak{A} \models \Phi' \text{ implies that } S^{\mathfrak{A}} \text{ is a total program,} \\ \text{then there exists a trivial schema } S_0 \text{ such that:} \\ \qquad \text{for all } \mathfrak{A},\ \mathfrak{A} \models \Phi' \text{ implies that } S_0{}^{\mathfrak{A}} \text{ defines the same function} \\ \text{on } A \text{ as } S^{\mathfrak{A}}. \end{cases}$$

This result may be used in many different ways. For example, one of Paterson's theorems states that *if a schema terminates under all interpretations then it must be equivalent to a loop-free schema* [see Paterson (1968)]. This fact is an immediate corollary of [$*$], by taking Φ to be the empty set.

Another less trivial consequence is the following. Let $\mathscr{K}$ be the class of structures $\{\mathfrak{B} \mid \mathfrak{A} \subseteq \mathfrak{B}\}$ or $\{\mathfrak{B} \mid \mathfrak{A} \preccurlyeq \mathfrak{B}\}$, for some fixed structure $\mathfrak{A}$, where $\subseteq$ and $\preccurlyeq$ mean "substructure" and "elementary substructure," respectively. (If $\mathfrak{A}$ is an elementary substructure of $\mathfrak{B}$ then the first-order properties of $\mathfrak{A}$ are preserved over to the superstructure $\mathfrak{B}$.) Let S be an arbitrary schema, which may be interpreted in every structure of $\mathscr{K}$. *$S^{\mathfrak{B}}$ is a total program for every $\mathfrak{B} \in \mathscr{K}$ if and only if there is a trivial schema S_0 such that, for every $\mathfrak{B} \in \mathscr{K}$, $S_0{}^{\mathfrak{B}}$ defines the same total function as $S^{\mathfrak{B}}$.* The proof of this fact is immediate from [$**$], because in both cases $\mathscr{K}$ is an EC_Δ class of structures, i.e., $\mathscr{K}$ is characterizable by a set Φ' of first-order sentences (see Bell and Slomson (1969), for example).

3. A Related Question

Given a program $S^{\mathfrak{A}}$, i.e., a program schema S in conjunction with a relational structure $\mathfrak{A}$ where S can be interpreted, can we formulate a set Ψ of first-order conditions such that, given any input $\mathbf{a}$ from A for $S^{\mathfrak{A}}$, Ψ is true of $\langle \mathfrak{A}, \mathbf{a} \rangle$ if and only if $S^{\mathfrak{A}}(\mathbf{a})$ terminates?

In view of the Representation Theorem for the partial recursive functions, it might be hoped that the answer to the above question is in the affirmative. For, given any program $S^{\mathfrak{N}}$ over the domain N of natural numbers and written in terms of the underlying relation and operations of $\mathfrak{N}$, the standard model of arithmetic (as one may readily verify, $S^{\mathfrak{N}}$ defines a partial recursive function), there exists a first-order existential sentence Ψ such that:

$$\text{for all } \mathbf{n} \in N, \langle \mathfrak{N}, \mathbf{n} \rangle \models \Psi \text{ if and only if } S^{\mathfrak{N}}(\mathbf{n}) \text{ terminates.}$$

This is not however the case in general. The structure $\mathfrak{M}$ is defined as $\langle M, =, (\ )', 0 \rangle$, where $(\ )'$ is the successor operation and M is the set N of natural numbers.

Proposition. *There is a program $S^{\mathfrak{M}}$ such that for no first-order sentence Ψ is it the case that*

$$\textit{for all } \mathbf{m} \in M, \langle \mathfrak{M}, \mathbf{m} \rangle \models \Psi \textit{ if and only if } S^{\mathfrak{M}}(\mathbf{m}) \textit{ terminates.}$$

Proof. (i) We first prove (in outline) that the first-order theory $Th(\mathfrak{M})$ of $\mathfrak{M}$ is decidable and that, given any first-order formula $\Psi(\mathbf{x})$, we can effectively find a first-order sentence Ψ_m such that: $\langle \mathfrak{M}, \mathbf{m} \rangle \models \Psi(\mathbf{m}) \Leftrightarrow \mathfrak{M} \models \Psi_m$, for all $\mathbf{m} \in M$. For this, we define a countable set of axioms, which are all true of $\mathfrak{M}$, then take their deductive closure T (the not too difficult construction of T is left to the reader). Clearly, we have that $T \subseteq Th(\mathfrak{M})$. We then show that T is $\aleph_1$-categorical, i.e., all the models of T which are of cardinality $\aleph_1$ are isomorphic (the proof of this simple fact is also left to the reader). Hence T is complete and so $T = Th(\mathfrak{M})$. Since T is also axiomatizable, $Th(\mathfrak{M})$ must be decidable, which proves the first part of the opening statement. As for the second part, it follows from the fact that any element of M other than 0 can be expressed as 0 to which $(\ )'$ has been applied finitely many times.

(ii) It is easy to verify that any partial recursive function can be defined by a flow-chart written in terms of $=$, $(\ )'$, and 0, i.e., by a program of the form $S^{\mathfrak{M}}$.

(iii) There are r.e. but not recursive predicates. Choose such a predicate, X, and the corresponding program, $S_x^{\mathfrak{M}}$, of which it is the domain. This is possible by (ii). Suppose there is a first-order sentence Ψ_x such that, for all $\mathbf{m} \in M$,

$$S_x^{\mathfrak{M}}(\mathbf{m}) \text{ terminates} \Leftrightarrow \langle \mathfrak{M}, \mathbf{m} \rangle \models \Psi_x ,$$

which is decidable by (i). But this contradicts the fact that, for all $\mathbf{m} \in M$,

$$S_x^{\mathfrak{M}}(\mathbf{m}) \text{ terminates} \Leftrightarrow \mathbf{m} \in X,$$

which is undecidable by the nonrecursiveness of X.

4. The Main Theorem

We establish a result which is somewhat stronger than is desired to answer the main question. The assertion [*] is an immediate consequence of the theorem below, since any family of interpretations

$$\mathscr{K} = \{\langle \mathfrak{A}, \mathbf{a} \rangle \mid \langle \mathfrak{A}, \mathbf{a} \rangle \models \Phi\},$$

for some set Φ of first-order sentences, is closed under ultraproducts. This fact is proved in Bell and Slomson (1969), which is our reference for all model-theoretic concepts.

Let S be an arbitrary schema. Let $\{\langle \mathfrak{A}_i , \mathbf{a}_i \rangle \mid i \in I\}$ be a family of interpretations for S, and D be an ultrafilter over the index set I. A *terminating path* through S, which goes from its input to its output in a direction prescribed by its arrows, is defined in the obvious sense. Observe that, along a terminating path, appear only finitely many instructions. We define the *ultraproduct of the computations* $\{S^{\mathfrak{A}_i}(\mathbf{a}_i) \mid i \in I\}$ relative to D as follows:

$$\prod \{S^{\mathfrak{A}_i}(\mathbf{a}_i) \mid i \in I\}/D = \begin{cases} \text{terminates, if there is a terminating path } \pi \\ \quad \text{through } S \text{ such that } \{i \mid S^{\mathfrak{A}_i}(\mathbf{a}_i) \text{ follows } \pi\} \in D \\ \text{diverges, otherwise.} \end{cases}$$

Whenever an ultraproduct of computations terminates, then its value is well-defined, i.e., in the above definition there can be no distinct terminating paths π_1 and π_2 through S such that the corresponding sets of indices are both members of D (index i belonging to the set corresponding to path π according to whether $S^{\mathfrak{A}_i}(\mathbf{a}_i)$ follows π).

LEMMA. *Let* $\{\langle \mathfrak{A}_i, \mathbf{a}_i\rangle \mid i \in I\}$ *be a family of interpretations for schema* S, *and* D *be an ultrafilter over* I. *Let* $\langle \mathfrak{B}, \mathbf{b}\rangle$ *be the ultraproduct*

$$\prod \{\langle \mathfrak{A}_i, \mathbf{a}_i\rangle \mid i \in I\}/D,$$

We then have

$$S^{\mathfrak{B}}(\mathbf{b}) = \prod \{S^{\mathfrak{A}_i}(\mathbf{a}_i) \mid i \in I\}/D,$$

i.e., "the computation on the ultraproduct is equal to the ultraproduct of the computations."

Proof (outlined). Let π be a terminating path through S. One can readily verify that there is a formula $P(\mathbf{x})$ such that, given any interpretation $\langle \mathfrak{A}, \mathbf{a}\rangle$ for S,

$$S^{\mathfrak{A}}(\mathbf{a}) \text{ follows path } \pi \Leftrightarrow \langle \mathfrak{A}, \mathbf{a}\rangle \models P(\mathbf{a}),$$

where $P(\mathbf{x})$ is a finite conjunction of atomic and negative atomic formulas. To complete the proof, we show that

$$S^{\mathfrak{B}}(\mathbf{b}) \text{ follows } \pi \Leftrightarrow \{i \mid S^{\mathfrak{A}_i}(\mathbf{a}_i) \text{ follows } \pi\} \in D.$$

Equivalently, by the previous remark, we have to show that

$$\langle \mathfrak{B}, \mathbf{b}\rangle \models P(\mathbf{b}) \Leftrightarrow \{i \mid \langle \mathfrak{A}_i, \mathbf{a}_i\rangle \models P(\mathbf{a}_i)\} \in D.$$

This last double implication follows from Łoś's theorem, see Bell and Slomson (1969).

THEOREM. *Let* S *be an arbitrary schema. If* $\mathscr{K}$ *is a family of interpretations for* S *which is closed under ultraproducts, then*

$$\textit{for all } \langle \mathfrak{A}, \mathbf{a}\rangle \in \mathscr{K},\ S^{\mathfrak{A}}(\mathbf{a}) \textit{ terminates}$$

if and only if there exists a trivial schema S_0 *such that*

$$\textit{for all } \langle \mathfrak{A}, \mathbf{a}\rangle \in \mathscr{K}, \qquad S^{\mathfrak{A}}(\mathbf{a}) = {S_0}^{\mathfrak{A}}(\mathbf{a}).$$

Proof. The implication from right to left is immediate. For the opposite implication, we assume that (i) there is no trivial schema S_0 such that, for all $\langle \mathfrak{A}, \mathbf{a}\rangle \in \mathscr{K}$, $S^{\mathfrak{A}}(\mathbf{a}) = {S_0}^{\mathfrak{A}}(\mathbf{a})$, (ii) $S^{\mathfrak{A}}(\mathbf{a})$ terminates for every $\langle \mathfrak{A}, \mathbf{a}\rangle \in \mathscr{K}$,

and we shall get a contradiction. Under conditions (i) and (ii), it is easy to see that infinitely many terminating paths through S must be used by the interpretations in $\mathscr{K}$; i.e., there is a countably infinite number of terminating paths through S, say $\{\pi_i \mid i \in \omega\}$, and a countably infinite family of interpretations from $\mathscr{K}$, $\{\langle \mathfrak{A}_i , \mathbf{a}_i \rangle \mid i \in \omega\}$, such that

$$\text{computation } S^{\mathfrak{A}_i}(\mathbf{a}_i) \text{ follows path } \pi_i \text{ through } S,$$

for all $i \in \omega$. Consider any (nonprincipal) ultrafilter D over ω. Let $\langle \mathfrak{B}, \mathbf{b} \rangle$ be the ultraproduct $\prod\{\langle \mathfrak{A}_i , \mathbf{a}_i \rangle \mid i \in \omega\}/D$. Since no two computations from $\{S^{\mathfrak{A}_i}(\mathbf{a}_i) \mid i \in \omega\}$ follow the same terminating path and since D contains only infinite sets (because D is nonprincipal), the ultraproduct of the computations $\{S^{\mathfrak{A}_i}(\mathbf{a}_i) \mid i \in \omega\}$ relative to D is diverging, by definition. Hence, $S^{\mathfrak{B}}(\mathbf{b})$ is also diverging, by the lemma. On the other hand, $\mathscr{K}$ being closed under ultraproducts, $\langle \mathfrak{B}, \mathbf{b} \rangle \in \mathscr{K}$. Hence, $\mathscr{K}$ contains an interpretation $\langle \mathfrak{B}, \mathbf{b} \rangle$ such that $S^{\mathfrak{B}}(\mathbf{b})$ does not terminate, contradicting (ii).

5. Concluding Remarks

(1) As mentioned earlier, only flow-chart schemas have been explicitly considered in the above results. These results are however valid for any class of program schemas which contains nontrivial members (a nontrivial program schema being one which has infinitely many consistent, i.e., "usable by some interpretation," terminating paths). This includes all classes which have been defined in the literature—whether it is that of recursive program schemas, or flow-charts with counters, or loop program schemas, etc.

(2) The main result [$*$] can be proved directly using a compactness argument, without the need to introduce ultraproducts and ultraproducts of computations [see Kfoury (1973)]. We have preferred to introduce these concepts for reasons of conciseness and elegance—in particular, in order to minimize syntactic considerations.

(3) Our results can be entirely transposed into mathematical logic. We briefly point out here how this transposition goes. Flow-chart schemas are equivalent to a certain class of formulas in the logic $L_{\omega_1\omega}$, but they cannot be viewed as first-order objects [see for example Engeler (1968)]. The same is true of all nontrivial program schemas, whether flow-chart or not. Let $\mathscr{S}$ be the class of formulas of $L_{\omega_1\omega}$ which correspond to all program schemas. It is easy to see that trivial schemas correspond to first-order quantifier-free formulas. Assertion [$*$] can be rephrased as follows: Given any $\sigma \in \mathscr{S}$, and

any set Φ of first-order sentences, if $\Phi \vdash \sigma$, then there exists a first-order quantifier-free formula σ_0 such that $\Phi \vdash \sigma \leftrightarrow \sigma_0$. The proposition and the theorem, in Sections 3 and 4, can also be rephrased accordingly.

RECEIVED: June 18, 1974; REVISED: March 7, 1975

REFERENCES

BELL AND SLOMSON (1969), "Models and Ultraproducts," North Holland, Amsterdam.

ENGELER, E. (1968), "Formal Languages: Automata and Structures," Markham Publishing Company, Chicago.

KFOURY, A. J. (1973), Comparing algebraic structures up to algorithmic equivalence, *in* "Proceedings of IRIA Symposium on Automata, Languages, and Programming, July 1972" (M. Nivat, Ed.), North Holland, Amsterdam.

LUCKHAM, PARK, AND PATERSON (1970), On formalised computer programs, *Journal of Computer and System Sciences*, 4.

PATERSON, M. (1968), Program schemata, *in* "Machine Intelligence," (Michie, Ed.), Vol. 3, Edinburgh University Press, Edinburgh.